Community Detection in Networks

Prashant A. Jayannavar

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India

Community Detection in Networks

Thesis submitted in

May 2013

to the department of

Computer Science and Engineering

of

National Institute of Technology Rourkela

in partial fulfillment of the requirements

for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Prashant A. Jayannavar

[Roll: 109CS0148]

under the guidance of

Prof. Banshidhar Majhi

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India

Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, India. www.nitrkl.ac.in

Dr. Banshidhar Majhi
Professor

May 09, 2013

Certificate

This is to certify that the work in the project entitled *Community Detection in Networks* by *Prashant A. Jayannavar* is a record of his work carried out under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of *Bachelor of Technology* in *Computer Science and Engineering*.

Banshidhar Majhi

Acknowledgment

I take this opportunity to express my gratitude and regards to my guide Prof. Banshidhar Majhi for his exemplary guidance, monitoring and constant encouragement throughout the course of this project.

I also take this opportunity to express a deep sense of gratitude to my friends for their support and motivation which helped me in completing this task through its various stages.

I am obliged to the faculty members of the Department of Computer Science & Engineering at NIT Rourkela for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment. In particular I am thankful to Prof. B. K. Patra for offering advice and help on important issues relating to the project.

Lastly, I thank my parents for their constant encouragement without which this assignment would not have been possible.

Prashant A. Jayannavar

Abstract

An important property of networks/graphs modeling complex systems is the property of community structure, in which nodes are joined together in tightly knit groups (communities or clusters), between which there are only looser connections. The problem of detecting and extracting communities from such graphs has been the subject of intense investigations in recent years. This problem is very hard and not yet satisfactorily solved. In this project we explore and work on this community detection problem. We frame the problem as an optimization problem and hence explore the use of Genetic Algorithms (GAs) in solving the same. We have studied, analyzed and implemented several existing algorithms including standard ones and GA-based ones. The standard algorithms include the Girvan-Newman Algorithm and the Label Propagation Algorithm by Raghavan et al. while the GA-based one is Tasgin et al.s algorithm. We have also designed a new GA-based algorithm for the problem. We present a comparative performance (accuracy + efficiency) analysis of these algorithms (new + existing) to gain insights into the problem and reveal the advantages of our proposed algorithm over existing algorithms. We have also created some artificial datasets (based on standard existing algorithms like the one for LFR graphs) for the purpose of the analysis and have acquired some real-world datasets (like Zacharys karate club network, Lusseau's network of bottlenose dolphins, etc.) too.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Overview

The modern science of networks has pushed forward our understanding of complex systems. There are now growing interests in modeling large complex systems as networks (i.e. graphs), such as the World Wide Web (WWW), social networks, co-citation networks and biological networks. In these networks/graphs, each node is an entity, e.g. web page, person, or paper, and each edge indicates a relationship between two nodes, e.g. web link, co-cited, and protein interaction. One of the most significant features of such graphs/networks is community structure, or clustering, i.e. the organization of vertices in clusters, with many edges joining vertices of the same cluster and comparatively few edges joining vertices of different clusters. Such clusters/communities, can be considered as fairly independent compartments of a graph, playing a similar role like, e.g., the tissues or the organs in the human body. Figure 1.1 depicts a typical graph with community structure. Detecting communities is of great importance in sociology, biology, computer science, etc., disciplines where systems are frequently represented as graphs. This problem is very hard and not yet satisfactorily solved, despite the huge effort of a large interdisciplinary community of scientists working on it over the past few years. [1]

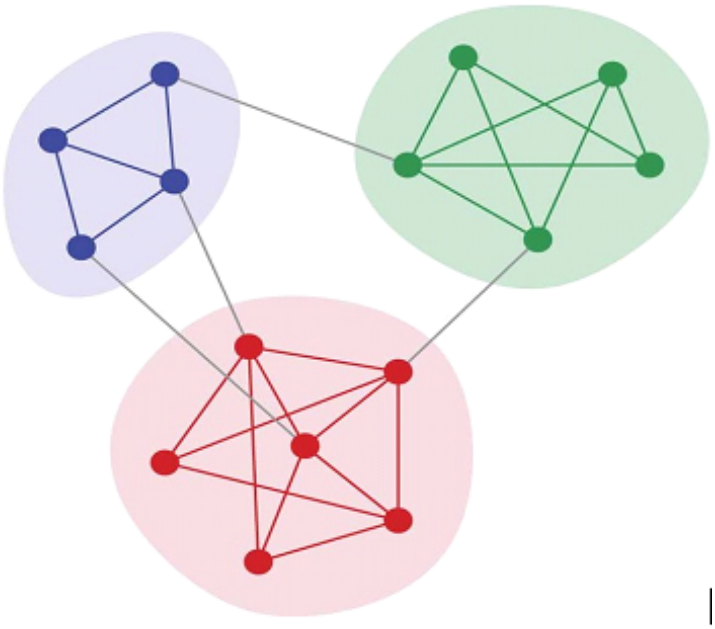

Figure 1.1: An example graph with 3 communities

Community detection in large networks manifests itself as two distinct problems:

- Global Community Detection: Given a network, detect or extract all communities
- Local Community Detection: Given a seed node in the network, identify the best community structure that includes this node, if there exists one

Several algorithms have been designed and tested for both problems. These include the Girvan-Newman algorithm, Radicchi et al.s algorithm, Raghavan et al.s algorithm, the Clauset-Newman-Moore algorithm, etc. for the former problem and Clausets algorithm, the CZR algorithm, Bagrows algorithm, etc. for the latter.

1.2 Applications and Importance

The community detection problem has many widespread applications and has hence proven to be very important. Clustering Web clients having like interests and who are in close geographical proximity to each other can improve the quality of services provided on the Web. Each community of clients can then be served by a dedicated mirror server. Identifying the community of a seed node in the co-purchase network of

on-line retailers (like Amazon) enables setting up efficient recommendation systems, that more effectively guide customers through the retailer's lists of items and enhance business opportunities. This happens to be an application of the local community detection problem. Ad hoc networks usually have no centrally maintained routing tables which specify the way of communication between nodes. Grouping the nodes into communities allows generating compact routing tables while the choice of the communication paths is still efficient.[1] Figure 1.2 shows community structure in a political books network. Detecting communities here helps to cluster the books into categories.

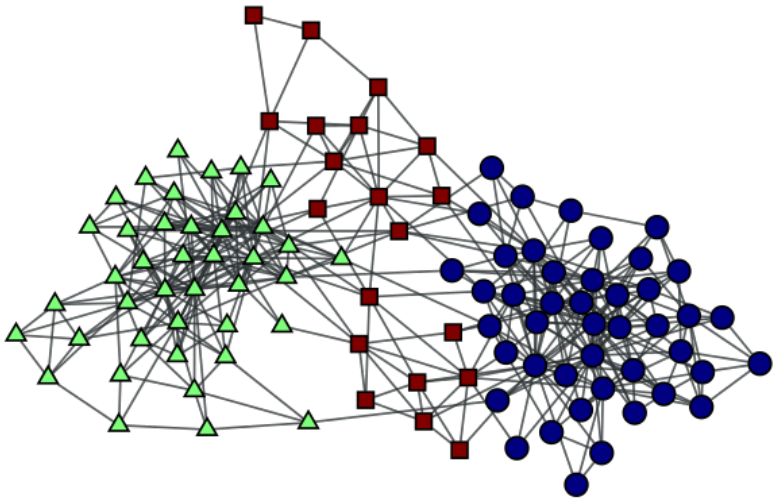

Figure 1.2: Community structure in a political books network

Community detection is important for other reasons, too. Proteinprotein interaction (PPI) networks are hugely important in biology and bioinformatics, because the interactions among proteins are fundamental for each process in a cell. Fig. 1.3 depicts a PPI network of the rat proteome. The proteins interact very often with each other, as they belong to metastatic cells, which have a high motility and invasiveness compared to normal cells. Clusters correspond to functional groups, i.e. to proteins

having the same/similar functions, which are expected to be involved in the same processes. Most of the communities are associated with cancer and metastasis. This indirectly shows how critical detecting clusters in PPI networks is. [1]

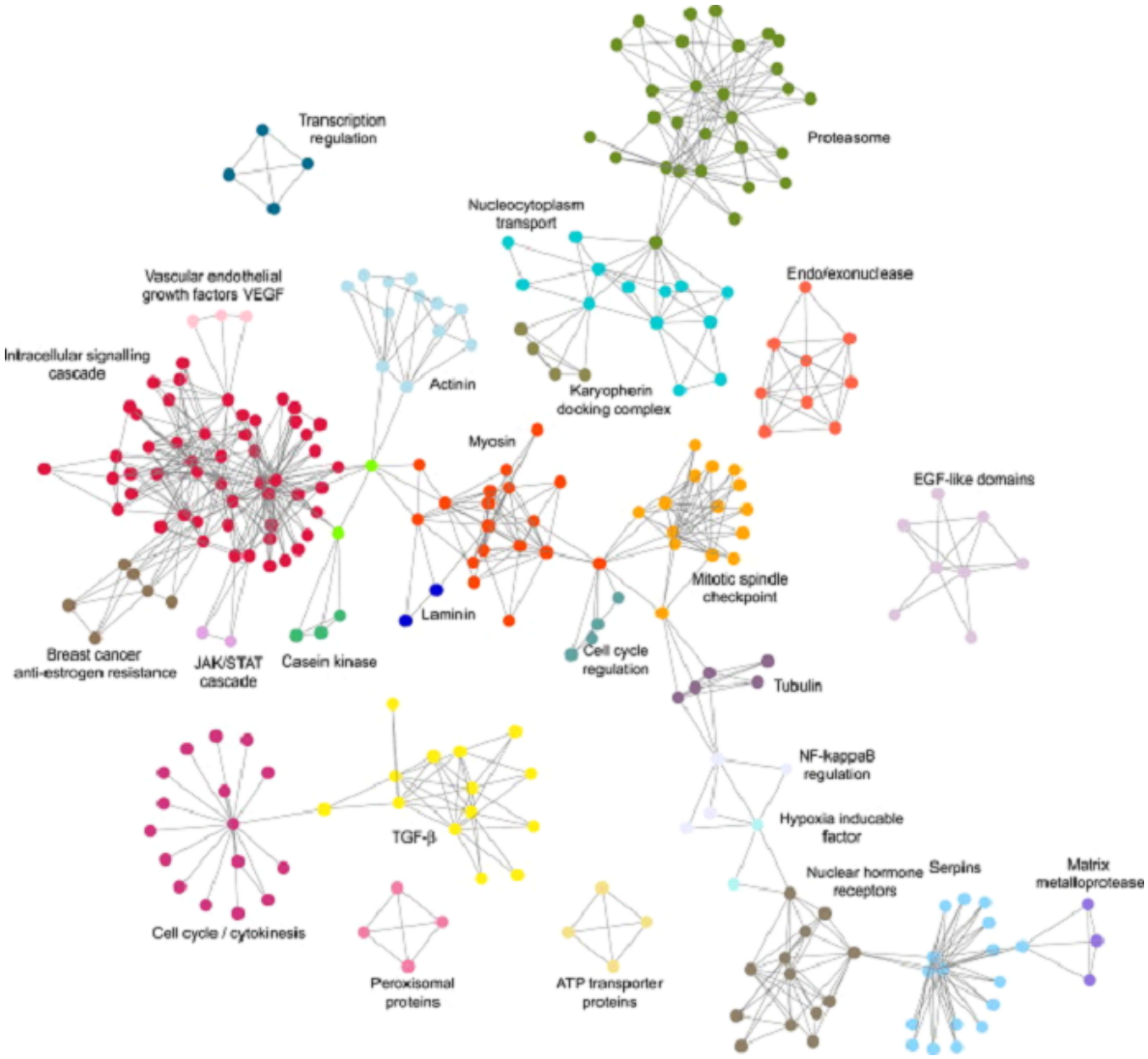

Figure 1.3: Community structure in PPI networks. The graph shows the interactions between proteins in cancerous cells of a rat.

Detecting communities in graphs is an important topic in computer science also. In parallel computing, for example, it is critical to know the best way of allocating tasks to processors so as to minimize the amount of inter process communication and achieve better speedups for parallel programs. This can be achieved by splitting the task/channel graph into communities and allocating tasks/processes belonging to the same community to a single node/processor in the computer cluster. [1]

1.3 Project Overview

In this project we explore and work on the first problem. We frame the problem as an optimization problem and hence explore the use of Genetic Algorithms (GAs) in solving the same. We have studied, analyzed and implemented several existing algorithms including standard ones and GA-based ones. The standard algorithms include the Girvan-Newman Algorithm and the Label Propagation Algorithm by Raghavan et al. while the GA-based one is Tasgin et al.s algorithm. The GN Algorithm was chosen as it happens to be one of the most accurate and the Label Propagation algorithm was chosen as it happens to be one of the fastest. Tasgin et al.'s algorithm was chosen as it is a GA-based one and very famous too. Such choices would suit a proper comparative study. We have also designed a new GA-based algorithm for the problem. We present a comparative performance (accuracy + efficiency) analysis of these algorithms (new + existing) to gain insights into the problem and reveal the advantages of our proposed algorithm over existing algorithms. We compare efficiency using running time as the metric and accuracy using network modularity as the metric. We have also created some artificial datasets (based on standard existing algorithms like the one for LFR graphs) for the purpose of the analysis and have acquired some real-world datasets (like Zacharys karate club network, Lusseau's network of bottlenose dolphins, etc.) too.

1.4 Thesis Organization

The rest of the thesis is organized as follows.

In **Chapter 2** the elements of community detection namely computational complexity, communities and partitions are described in relative detail. Brief descriptions of existing algorithms for the global problem are also presented.

Chapter 3 describes the proposed GA-based algorithm. It includes details of all components of the GA like encoding, fitness function, operators, etc.

The comparative performance analysis of the algorithms is presented in **Chapter 4**. Accuracy comparisons on real-world and artificial graphs and an efficiency comparison on real-world graphs are shown.

Finally **Chapter 5** presents the concluding remarks, with scope for further research work.

Chapter 2

Literature Review

2.1 Elements of Community Detection

The elements of community detection have been well documented by Fortunato [1]. The three primary elements of community detection are: Computational complexity, Communities and Partitions.

2.1.1 Computational Complexity

Efficiency is a critical issue for clustering algorithms because of the enormous amount of data on current real-world networks. The computational complexity of an algorithm is an estimate of the amount of resources (time and space) required by the algorithm to perform its specific tasks. Time taken is estimated by the number of computation steps performed by the algorithm and space consumed is estimated by the number of memory units that are needed by the algorithm. Expressing the scalability of these demands with the size of the problem being studied is a standard technique for analyzing algorithms. In dealing with a graph, the size is expressed by the number of nodes n and/or the number of edges m.

Many clustering algorithms or problems related to clustering are NP-hard. This means that it is futile to use exact algorithms for obtaining the solution, which could be used only for very small systems. Also, even if an algorithm has a polynomial complexity, it may still be too slow to actually practically work for large systems of

interest. In all such cases it is common to use approximation algorithms. These are algorithms that produce an approximate solution instead of an exact one, with the benefit of a lower complexity. The goal is to deliver a solution which differs by a constant factor from the optimal solution. Approximation algorithms are very often used for optimization problems, in which one wants to find the maximum or minimum value of a given cost function over a large set of possible system configurations.

2.1.2 Communities

The first issue in community detection is to look for a quantitative definition of a community. No definition is universally accepted. In fact, the definition frequently depends on the system at hand and/or the application one has in mind. An intuitive idea is that there must be more edges inside the community than edges linking nodes of the community with the rest of the graph. This notion is at the basis of most community definitions. But other alternative formulations are also possible. Also, in most cases, communities are algorithmically defined, i.e. they just happen to be the final product of the algorithm, without a concrete a priori definition.

Many local definitions (e.g. clique, strong community, weak community, etc.), global definitions (e.g. ones based on modularity, etc.) and definitions based on vertex similarity (e.g. ones based on euclidean distance, cosine similarity, etc.) have been proposed.

2.1.3 Partitions

A partition is a division of a graph in communities, such that each vertex belongs to one community. In real systems, however, a vertex may belong to multiple communities (e.g. a person can belong to multiple social circles in a social network). A division of a graph into overlapping communities is called a cover. Partitions can be hierarchically ordered, when the graph has different levels of organization at different scales. In such a case, clusters in turn display community structure, with smaller clusters inside, which may again contain smaller clusters, and so on.

Many clustering algorithms are able to identify multiple meaningful partitions. As not all the partitions are equally good, it is helpful to have a quantitative criterion to

judge the goodness of a partition. A quality function is a function that assigns a value (a number) to each partition of a graph. The number is a measure of the goodness of the partition. We can then rank partitions based on their quality function value. Nevertheless, it should be kept in mind that the question of whether a partition is better than another one is ill- posed, and the answer depends on the specific concept of community considered and/or quality function used.

Many quality functions for determining the quality of a partition have been proposed, the most famous one being Network Modularity proposed by Girvan and Newman [2].

2.2 Existing Algorithms

One of the most famous algorithms for the problem has been presented by Girvan and Newman in a seminal paper published in 2002 [2]. The method is very accurate and is a divisive hierarchical clustering method based on an iterative removal of edges from the network. The network is gradually split into communities by the edge removal process. Betweenness measures are used to determine the order of removal of edges. The algorithm uses Freeman's betweenness centrality [3] (extended for edges) as its betweenness measure. The idea behind the edge betweenness comes from the observation that if two communities are joined by a few inter-community edges, then all the (shortest) paths from nodes in one community to nodes in the other must pass through these edges. These shortest paths determine the betweenness centrality scores of the edges. Counting all the shortest paths passing through each edge gives us the betweenness score of each edge. The edge with the maximum score is then removed. The scores are then recalculated and the process is repeated, thus dividing the network into smaller components until a stop criterion is reached. The stopping criterion used is the network modularity defined by Girvan and Newman. Thus the algorithm computes the modularity of all the partitions obtained from the hierarchical approach, and returns the partition having the maximum modularity value. Brandes' algorithm [4] can be used to compute the betweenness centralities very efficiently.

One of the fastest (near linear time) algorithms for the problem is Raghavan et al.'s Label Propagation algorithm [5]. It is a simple label propagation algorithm that

requires no optimization of a predefined objective function. Every vertex is initialized with a unique label and at every step each vertex adopts the label that most of its neighbors currently have. In this iterative process densely connected groups of nodes form a consensus on a unique label to form communities.

Tasgin et al. have proposed a GA-based algorithm for the problem [6]. They show that the employment of genetic algorithms for the community detection problem is a viable approach and propose an algorithm which tries to optimize network modularity using GA methods.

Chapter 3

Proposed Work

3.1 Role of Optimization

The community detection problem is basically a search for the optimal partition of a graph. It can therefore be framed as an optimization problem with a search space consisting of all possible partitions of a graph. The number of possible partitions in k clusters of a graph with n vertices is the Stirling number of the second kind *S(n, k)* [7]. The total number of possible partitions is then the n-th Bell number $B_n = \sum_{k=0}^{n} S(n,k)$ [7]. In the limit of large n, B_n has the asymptotic form [8]

$$B_n \sim \frac{1}{\sqrt{n}}[\lambda(n)]^{n+1/2}\mathrm{e}^{\lambda(n)-n-1} \tag{3.1}$$

where $\lambda(n) = e^{W(n)} = n/W(n)$, *W(n)* being the Lambert W function [9]. We thus see that B_n grows faster than exponentially with graph size n. Hence an enumeration and/or evaluation of all partitions is impossible, unless the graph consists of a very small number of nodes. It can hence be seen that the problem possesses a huge solution space which a GA would help in exploring efficiently and effectively. In case of a GA the fitness/objective function would then have to be a partition quality function. Maximizing the fitness function would then mean trying to find the best solution, i.e., partition of the graph.

3.2 The Proposed Algorithm

3.2.1 GA Framework

The proposed algorithm uses a genetic algorithm based approach. The GA framework (same as in [6]) used by us is as follows:

- Generate initial population (population size = p + βp)
- Bias the initial population
- In each of the gen iterations (generations) do:
  - Apply the fitness function to chromosomes
  - Sort the chromosomes w.r.t. the fitness value and take the top p
  - Save the top p chromosomes for later use
  - Pair the top p sorted chromosomes and apply crossover operation to the pairs
  - Apply mutation
  - Combine newly obtained p chromosomes and the previously saved βp

Here gen, β and p are model parameters.

3.2.2 Encoding

Given a graph G(V, E) with |V| = n and |E| = m, any partition P of G is represented by the following decimal encoding:

a vector k = [k_1 k_2 ... k_n]

where, k_i is the index of the cluster of vertex v_i

3.2.3 Generating Initial Population

Generate each chromosome vector by: k = [k_1 k_2 ... k_n] where, k_i = i (i.e. each node is in a cluster of its own). So, k = [1 2 ... n] .

3.2.4 Biasing the Initial Population

For each chromosome:

Do αn number of times:

Randomly choose a vertex v_i

Assign its cluster to all of its neighbors

(i.e. $k_j \leftarrow k_i$ whenever $(v_i, v_j) \in$ E)

Here α is a model parameter.

3.2.5 Fitness Function

We define a fitness function (that has also simultaneously been defined by Chen et al. [10]). It measures the community quality.

$$Z = \text{average internal degree/average external degree} \tag{3.2}$$

$$Z_C = \frac{Z_{in}}{Z_{ex}} \text{ (C=community)}$$

where,

$$Z_{in} = \frac{\sum_{i \in C} indeg(i)}{|C|} \tag{3.3}$$

$$Z_{ex} = \frac{\sum_{j \in B} outdeg(j)}{|B|} \tag{3.4}$$

$indeg(i)$ = number of neighbors of i in C

$outdeg(j)$ = number of neighbors of j not in C

B = boundary of the community

= set of nodes of C having at least one neighbor outside C

Z was now be extended by us to measure the partition quality too.

$$Z_{partition} = \sum_{c} Z_c \tag{3.5}$$

3.2.6 Crossover Operator

[6] defines a one-way crossover operator. We define a one-way 'biased' crossover operator as follows: Let p_1 be the source chromosome and p_2 be the destination chromosome. If $p_1 \rightarrow p_2$ then, we copy some of the best clusters from p1 to p2 (rather than some random ones). Similarly, $p_2 \rightarrow p_1$ is also possible

Here, the crossover rate μ is a model parameter.

3.2.7 Mutation Operator

We define a 'biased' point mutation operator as follows:

- Randomly choose a community C
- Randomly choose a boundary node v of C
- Calculate belongingness of v to C (b_0) and other communities $C_1, C_2, ...(b_1, b_2, ...)$ to which it is linked
- Remove v from C and add it to that C_i for which v possesses the largest $b_i > b_0$

We define belongingness of a node v to cluster C as:

$$B_v = \frac{indeg(v)}{deg(v)} - \frac{outdeg(v)}{deg(v)} \tag{3.6}$$

Here, the mutation rate τ is a model parameter.

3.3 Comments on the Proposed Work

3.3.1 Role of biasing the GA

In the design of our algorithm, we have given a lot of importance to 'biasing' the genetic algorithm. Biasing is a way to reduce the arbitrariness / randomness of a GA by using more intuitive approaches and heuristics. It helps improve the speed of convergence of the GA. Biasing techniques used should not result in significant computational overheads, otherwise, the overall running time may increase even if the number of iterations required to converge may decrease. Apart from improving efficiency, biasing may also help in improving the accuracy of the algorithm as it uses more intuitive techniques and heuristics which may prove to perform better than techniques based on randomness.

3.3.2 Similarities and Differences with Existing Algorithms

The proposed algorithm is similar to Tasgin et al.'s GA-based algorithm in the flowing aspects:

1. The GA framework
2. Encoding
3. Generating and biasing the initial population

The novelties of the proposed algorithm which are different from Tasgin et al's GA-based algorithm are:

1. The fitness function
2. The crossover operator
3. The mutation operator

Chapter 4

Results and Analysis

In this chapter we present a comparative performance analysis of the algorithms. We compare efficiency using running time as the metric and accuracy using network modularity as the metric. [Note: For the purpose of the comparative analysis, the model parameters for the proposed algorithm have been set to the following values: $\beta = 0.1, p = 100, \alpha = 0.4, \mu = 0.2, \tau = 0.5$]

4.1 Datasets

The real-world datasets used include graphs like Zachary's karate club network, Lusseau's network of bottlenose dolphins, the NCAA football network, the Jazz network and the Elegans network. Figures 4.1 and 4.2 depict Zachary's network and Lusseau's network respectively.

Dataset	Nodes	Edges
Zachary's Karate Club	34	78
Lusseau's Dolphins	62	159
Jazz	198	5485
Elegans	453	4597

Table 4.1: Graph sizes of the real-world datasets

The artificial/synthetic datasets used include LFR graphs proposed by Lancichinetti et al. [11]. 6 different graphs for 6 values of the LFR parameter ranging from 0.1

through 0.6 were created for the purpose of the analysis.

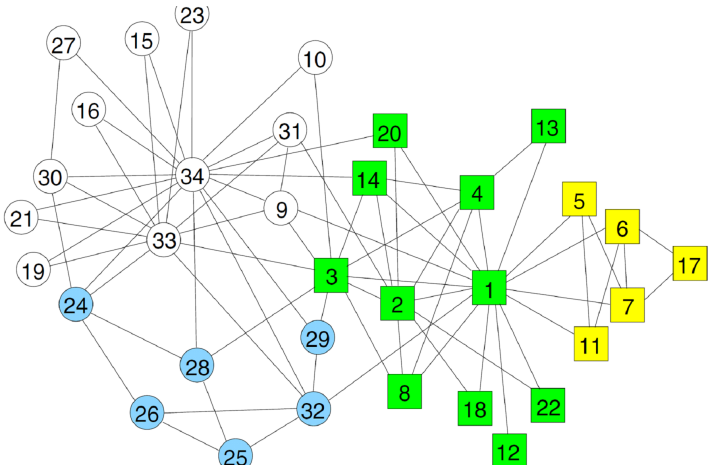

Figure 4.1: Zachary's Karate Club Network

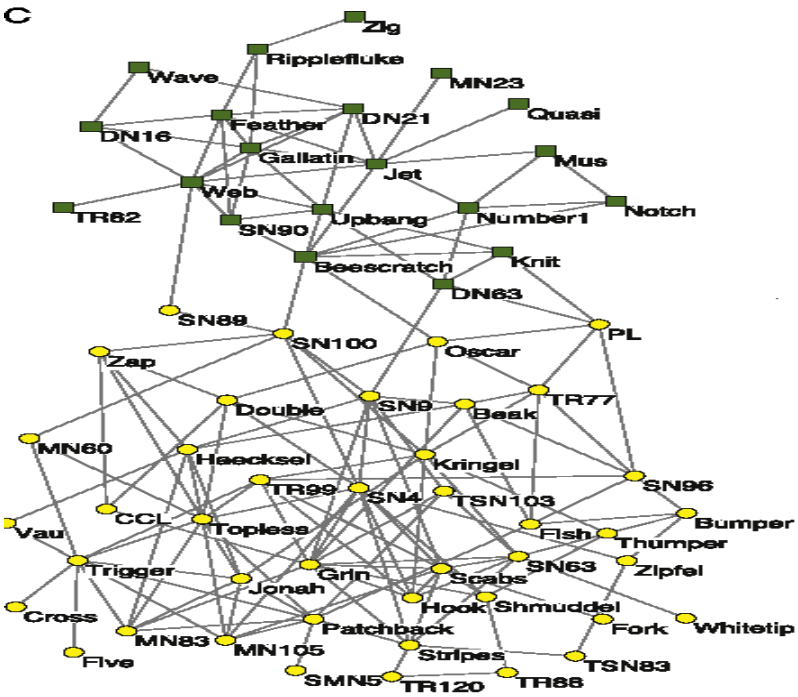

Figure 4.2: Lusseau's network of bottlenose dolphins

4.2 Comparision on Real World Graphs

4.2.1 Efficiency Comparision

The running times of the algorithms were compared on all of the real world graphs. The metric chosen was running time. The following plot (depicted in figure 4.3) was obtained:

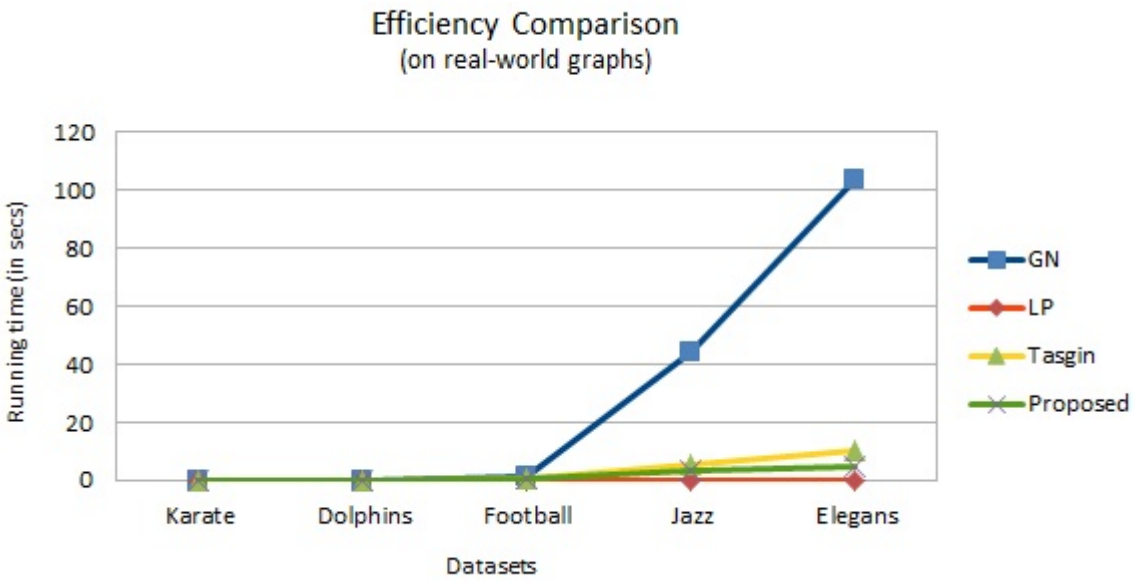

Figure 4.3: Plot of Running time v Datasets for the algorithms

The proposed algorithm turns out to be very fast, second only to Raghavan et al.'s Label Propagation algorithm.

4.2.2 Accuracy Comparision

The accuracies of the algorithms were compared on all of the real world graphs. The metric chose was Modularity. The following plot (depicted in figure 4.4) was obtained:

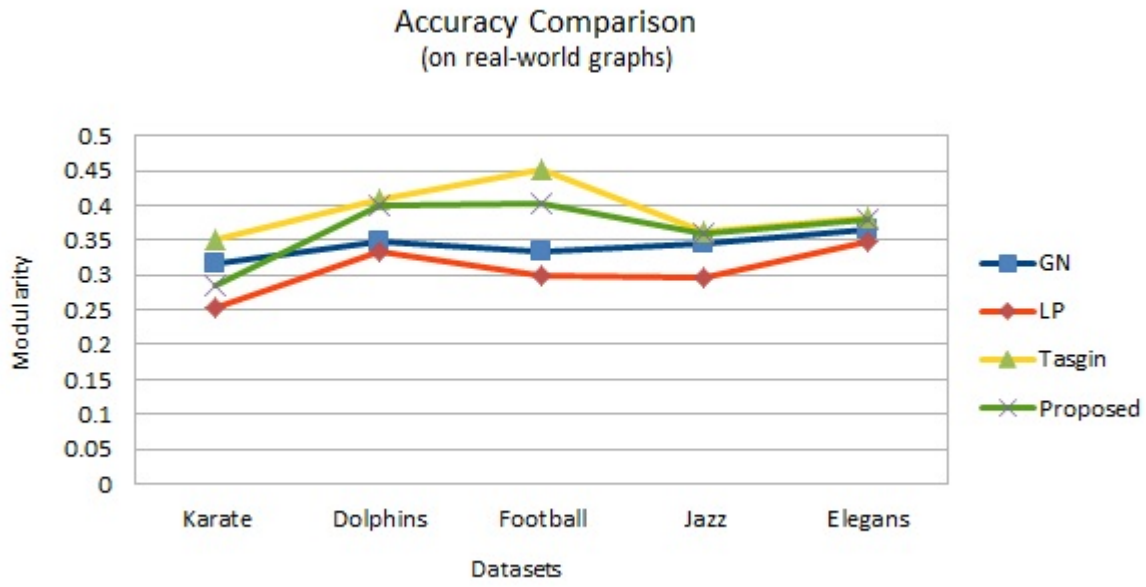

Figure 4.4: Plot of Modularity v Datasets for the algorithms

The proposed algorithm turns out to very accurate, second only to Tasgin et al.'s GA-based algorithm. Moreover, the latter explicitly optimizes modularity. Hence, its accuracy is understandably quite high when modularity is used as the metric for comparison.

4.3 Comparision on Artificial Graphs

The accuracies of the algorithms were compared on the six LFR graphs (with LFR parameter value ranging from 0.1 through 0.6). The metric chose was Modularity. The following plot (depicted in figure 4.5) was obtained:

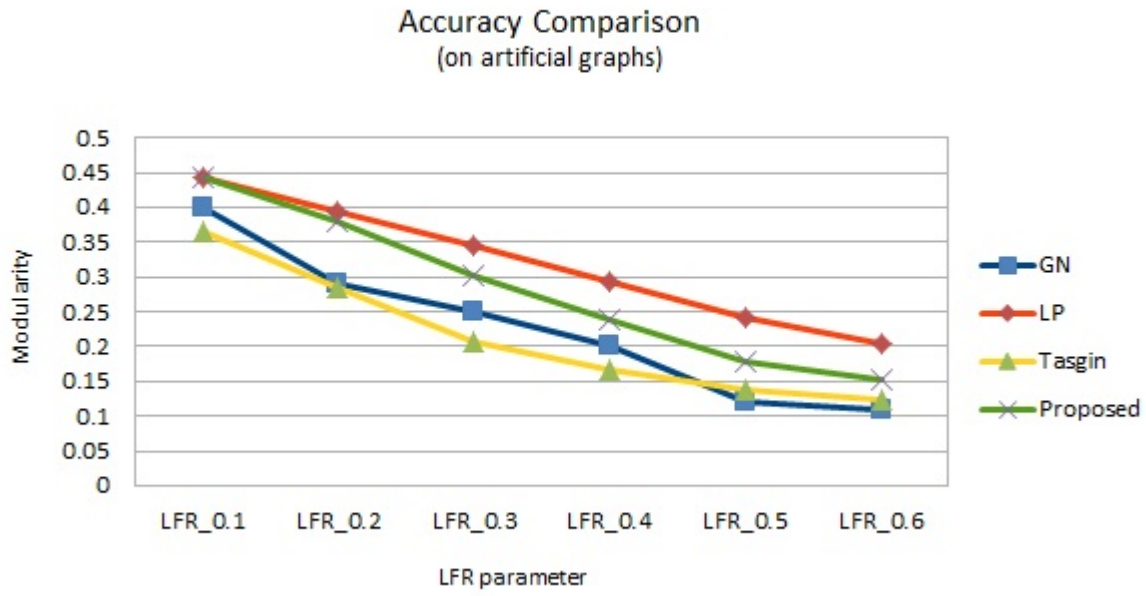

Figure 4.5: Plot of Modularity v LFR parameter for the algorithms

The proposed algorithm turns out to very accurate, second only to Raghavan et al.'s Label Propagation algorithm. The accuracies of all algorithms decrease gradually as the parameter value increases. This is because community structure in the LFR graph becomes increasingly less distinct and hence increasingly more difficult to extract.

Chapter 5

Conclusion

5.1 Contributions

This thesis proposes a new GA-based community detection algorithm for the global problem. It is similar to Tasgin et al.'s algorithm in that it borrows the same GA framework, encoding scheme and way of generating and biasing the initial population. The novelties of our algorithm, however, lie in the definitions of the fitness function, crossover operator and mutation operator. We have used the technique of 'biasing' in the design of our algorithm. The proposed algorithm has the following advantages over existing algorithms:

1. It provides a balance between speed and accuracy lacking in existing algorithms

2. It performs consistently well on both real-world and artificial datasets unlike existing algorithms

The second contribution is a comparative performance (accuracy + efficiency) analysis of all algorithms (new + existing) which gives us insights into the (global) problem of community detection and reveals the advantages of our proposed algorithm over other existing algorithms.

5.2 Future Work

Future research can aim at:

1. tuning the GA parameters for achieving peak performance
2. exploring new approaches for implementing GA components like encoding scheme, fitness function, operators, etc.
3. applying multi-objective GA-based approaches to the problem

Bibliography

[1] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[2] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[3] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[4] Ulrik Brandes. A faster algorithm for betweenness centrality*. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[5] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106, 2007.

[6] Mursel Tasgin, Amac Herdagdelen, and Haluk Bingol. Community detection in complex networks using genetic algorithms. *arXiv preprint arXiv:0711.0491*, 2007.

[7] George E Andrews. *The theory of partitions*, volume 2. Cambridge University Press, 1998.

[8] László Lovász. *Combinatorial problems and exercises*, volume 361. American Mathematical Soc., 1993.

[9] George Pólya, Gábor Szegő, and Dorothee Aeppli. *Problems and Theorems in Analysis I: Series, Integral Calculus, Theory of Functions*, volume 1. Springer Verlag, 1998.

[10] Jiyang Chen, Osmar Zaïane, and Randy Goebel. Local community identification in social networks. In *Social Network Analysis and Mining, 2009. ASONAM'09. International Conference on Advances in*, pages 237–242. IEEE, 2009.

[11] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):046110, 2008.